BOEING RESEARCH REPORT

A COMPARISON OF TREE SEARCH SCHEMES FOR DECISION NETWORKS *

W. B. Crowston and M. H. Wagner

March 1969 380-69

* The authors gratefully acknowledge the support received for this research from the Boeing Research Project, Sloan School of Management, Massachusetts Institute of Technology.

TABLE OF CONTENTS

INTRODUCTION

Decision CPM has been proposed by Crowston and Thompson [2] as a method for the simultaneous planning and scheduling of projects. In conventional CPM analysis, decisions are made as to the specific method for performing each task before the project is scheduled. However, decision CPM allows the planner to include all reasonable alternative methods of performing each job in the project graph, the alternatives possibly having different costs, different time durations, and different technological dependencies, and to then pick the best combination of job alternatives in light of cost and due-date considerations. In addition, interdependency between methods used to complete different tasks may be included. Such alternative interdependencies may arise from a requirement for consistency in the materials or designs used, for example.

Crowston and Thompson show how Decision CPM networks can be given a mathematical representation using integer variables. Crowston also shows in [1] that all nondecision jobs (i.e., jobs for which there are no alternatives) may be eliminated from the original network in a fashion so that resulting "reduced network" is equivalent in the following sense: Given a set of decisions, the resulting early start times for all decision jobs in the reduced network will be the same as those in the original. The optimal solution to the reduced network is thus optimal to the original.

The reduced problem is formulated as follows: associate with each job S_{ij} in the network its cost C_{ij}, its early start time w_{ij}, and an integer variable d_{ij} which takes the value 1 if the job is performed and 0 if it is not. The subscript "ij" indicates that a decision job is the jth alternative for decision node i. A due date DD is assumed with a premium of "r" dollars per time period of early completion and a penalty of "p" dollars per time period of late completion. Then find $D = \{ d_{ij} \}$ so as to minimize the sum of job cost and due-date penalty or premium, that is:

$$\text{Min } Z = \sum_{j=1}^{N} \sum_{j=1}^{k(i)} d_{ij} C_{ij} - r w_F^- + p w_F^+$$

subject to:

Interdependency

(a) Mutually exclusive:

$$\sum_{j=1}^{k(i)} d_{ij} = 1 \qquad l = 1, \ldots, N$$

(b) Alternative: as required i.e.

$$d_{ij} \leq d_{mn}$$

$$d_{ij} = d_{mn}$$

Precedence

$$- M(1 - d_{ij}) + w_{ij} + t_{ijkl} \leq w_{kl} \qquad M = "\infty"$$

where S_{ij} is an immediate predecessor of S_{kl}

Project Completion

$$w_F - w_F^+ + w_F^- - DD = 0 \qquad w_F^+, w_F^- \geq 0$$

where N is the number of decision nodes; k(i) is the number of alternatives for decision node (i); t_{ijkl} is the time required from the start of the predecessor S_{ij} to the start of its immediate successor S_{kl}; and w_F is the early start time of project finish.

For the solution procedure to be described, that is Branch and Bound, more general objective functions may be easily handled. For example, rewards and penalties may be associated with the completion of particular tasks in the project [1]. It is required, however, that Z be nondecreasing with completion time. The algorithms described in this paper are concerned with finding the optimal solution(s) in terms of Z. The problem of finding the project cost curve (minimum cost for each possible completion date) is only slightly more difficult and is discussed in [9].

THE BRANCH AND BOUND TECHNIQUE

The method of Branch and Bound has been adopted for the solution of DCPM networks. Branch and Bound, otherwise known as Combinatorial Programming or Controlled Enumeration, is an intelligently structured search of the space of all possible solutions. The procedure is based on two principal concepts: the use of a controlled enumeration technique for (implicitly) considering all potential solutions, and the elimination from explicit consideration of particular potential solutions which are known from bounding or feasibility considerations to be unacceptable. The technique has been applied to a variety of problems, most of which require all or mixed integer solutions.[1]

[1] For a survey of Branch and Bound applications, see [6].

The method as applied to this problem is illustrated by the tree search diagram of Figure 1. The solution space is first partitioned into the mutually exclusive collectively exhaustive subspaces of solutions defined by the choice of one of the possible alternatives for decision node 1. Then each of these subspaces is further partitioned using another decision node. If the process is continued through all decision nodes, every solution satisfying the mutually exclusive interdependency conditions is obtained. In Figure 1, the case where there are only two alternatives for each decision node is illustrated, but the extension to the case where there are k(i) alternatives for node i can be easily visualized.

A partial solution: p_n^k is a set of n decisions $d_{\alpha(m)j} = 1$ $m = 1, \ldots, n$ where k is some bookkeeping label and α is a vector of decision node labels i in some order. An augmentation of p_n^k corresponds to the choice of a job alternative for decision node $\alpha(n+1)$ to be done in conjunction with p_n^k

$$p_{n+1}^{k'} = p_n^k \, d_{\alpha(n+1)j}.$$

A completion: $p_N^{k'}$ of p_n^k results from a series of augmentations such that a decision is made for each node. In Figure 1, the labels have been chosen so that $p_3^7 = \{d_{1,1} ; d_{3,1}\}$ is an augmentation of $p_2^3 = \{d_{1,1} ; d_{2,1}\}$. If N = 4, p_4^6 and p_4^5 are two possible completions of p_2^3.

It is important to note that a partial solution p_n^k may define one or more paths from start to finish in the "reduced network" of the project. If so, a project completion date w_F^k is defined for the partial solution.

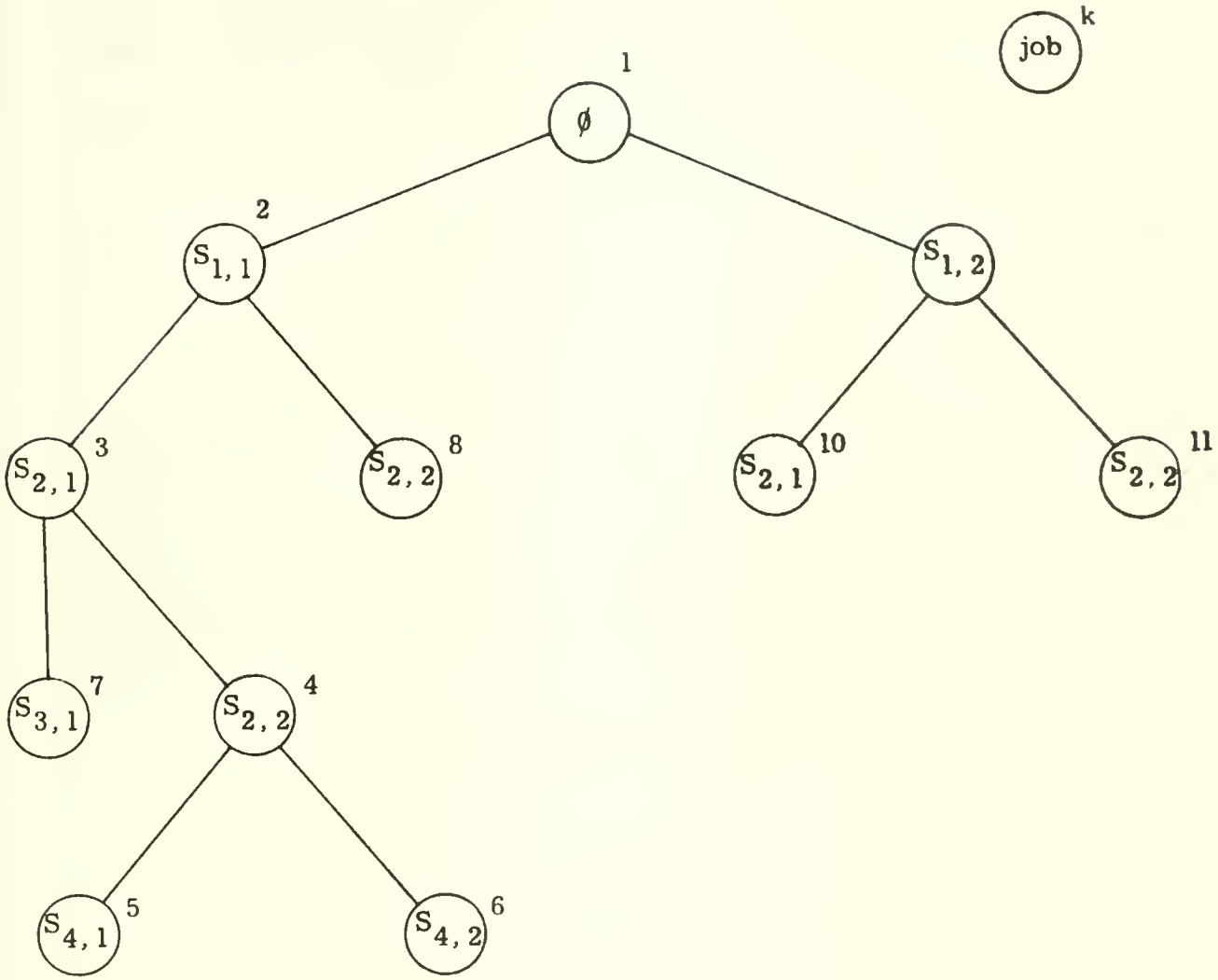


Figure 1. Branch and Bound Solution Tree.

Then w_F^k is a lower bound on the completion date which will result from any completion of p_n^k. Similarly, the quantity $C^k = \sum_{i=1}^{n} C_{ij}\, d_{ij}$ is a lower bound on the total job cost which will result from any completion of p_n^k. Then

$$Z^k = C^k + pw_F^{k+} - rw_F^{k-}$$

is a lower bound on the objective function which will be obtained from any completion of p_n^k.

Suppose some value Z^* is the value Z for the best complete solution found thus far. Then if $Z^* < Z^k$, p_n^k is bounded and need no longer be considered. Furthermore, all completions of p_n^k may be discarded. Similarly, if p_n^k can be shown to be infeasible from alternative interdependency considerations, then it and all of its possible completions may be discarded.

The Branch and Bound technique is a procedure for generating partial solutions by successive augmentations of the starting partial solution p_o^k (no decisions). At each stage, the current partial solution is checked to determine whether it is bounded or infeasible. If so, it is discarded, along with its subspace of possible completions, and another partial solution is considered. The process continues until the entire solution space has been explicitly or implicitly considered. The best feasible solution found at this point is optimal.

FIXED ORDER ALGORITHM

The algorithm described in this section is termed "fixed order" because the order α in which decision nodes are processed is predetermined and fixed. The essential features of the algorithm are presented in the

following order: feasibility tests, determination of bounds, branching and backtracking, and algorithm termination. A flow chart and sample problem solution follow, and then some considerations for choice of effective order α are discussed.

Feasibility Tests

The programmed version of this algorithm is equipped to handle pairwise interdependency constraints of the following types:

$$d_{ij} = d_{rs}$$
$$d_{ij} \neq d_{rs}$$
$$d_{ij} \leq d_{rs}$$

Tests for feasibility are made at each augmentation over interdependency constraints which involve decision node (n+1).

Determination of Bounds

A bound for a partial solution has two components: the cost for those job alternatives selected on the route, and the penalty or premium associated with the lower bound on w_F implied by the partial solution w_F^k.

Determination of cost is a straightforward additive procedure. It is important, however, to first normalize the costs associated with each decision node by subtracting the minimum cost from each of the decision job costs associated with that node.

$$C'_{ij} = C_{ij} - \min_j (C_{ij})$$

This normalization carried over all decision nodes yields a constant

representing the minimum total decision job cost which must be added to the objective function.

$$Z = \sum_{i} \min_{j} (C_{ij}) + \Sigma C'_{ij} \; d_{ij} + pw_F^+ - rw_F^-$$

This step tightens the cost portion of the lower bound for a partial solution by incorporating the fact that a complete solution must include, at least, the cost of the cheapest job alternative for each decision node not yet considered. Since the algorithm works with reduced networks, the cost of nondecision jobs is assumed to be zero. For real projects, this cost must be added to the objective function, but it is fixed cost and has no effect upon the workings of the algorithm.

The "completion time" portion of the bound is determined by a standard forward pass through the network defined by the partial solution. This is accomplished in the programmed version of the algorithm by setting $d_{ij} = 0$ for all decision jobs not selected by the partial solution, and then conducting the forward pass calculations over only those time constraints for which the d_{ij} for both predecessor and successor are positive. The method does not require distinguishing between those decision jobs which have not been selected because they have not yet been considered for the current partial solution, and those rejected in favor of another alternative job. The artificial start and finish jobs are assumed to have been selected by all solutions.

There are many cases in which the completion time portion of the bound for a partial solution p_n^k could be strengthened by taking into account decisions which must be made for decision nodes $\alpha(m)$, $m > n$. For example, in Figure 2 it can be seen that a choice of $S_{1,1}$ implies a lower

bound on completion time of six days, even though no path exists through the network until a decision is made for decision node 2. The algorithm does not recognize such cases; artificial nondecision jobs may be inserted in the network to represent these and more complicated situations, however.

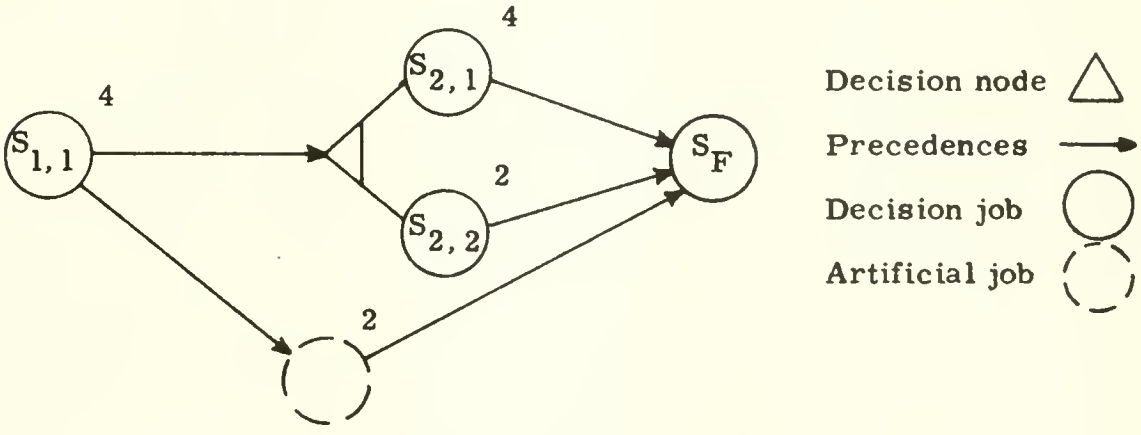


Figure 2.

Branching and Backtracking

An "iteration" for this algorithm refers to the process of picking some partial solution p_n^k for further elaboration, and then generating and evaluating each of the possible augmentations $p_n^k \cup d_{ij}$, $j = 1, \ldots k(i)$ where $i = \alpha(n + 1)$. The rule used to determine which of all active (feasible and unbounded) solutions to consider for the next interation is the following: choose the partial solution p_n^k which has the lowest value Z^k from the set of all active solutions most nearly complete. This rule is used for both "branching" (i.e., proceeding down the solution tree) and "backtracking" (i.e., proceeding back up the tree when the current partial solution is completed or discarded). It can be seen that this procedure amounts to picking the most promising of the most recently

generated active partial solutions and shall be referred to as a LIFO strategy. The LIFO strategy was adopted primarily because it tends to minimize computer storage requirements. The strategy is implemented by the maintenance of a pushdown list, γ, of solution labels k.

Algorithm Termination

The algorithm will terminate when all possible solutions have been considered explicitly or implicitly. Since there are a finite number of solutions, termination is guaranteed. The partitioning procedure is such that the entire space of potentially optimal solutions is spanned by the set of active partial and complete solutions. When the number of complete solutions which have the lowest Z value = Z* is equal to the number of active solutions, the algorithm terminates.

Flow Chart

The complete fixed order algorithm operates as follows after α is specified:

Step 1: $Z^* = \infty$

$\gamma = \{1\}$

$p_o^1 = \{\emptyset\}, \quad Z^1 = 0$

Step 2: $K = \gamma(1)$; current solution is p_n^k.

If $Z^k > Z^*$, go to Step 6.

If $Z^k \leq Z^*$, go to Step 3 if $n < N$;

Step 5 if $n = N$.

Step 3: $i = \alpha(n + 1)$

Step 4: Evaluate each of the augmentations $p_n^k \cup d_{ij}$ $j = 1, k(i)$ by testing each for:

a. feasible?

b. $Z^{k'} \leq Z^*$?

Save the solutions which pass these tests, and insert the new labels k' at the top of the list γ in order of nondecreasing $Z^{k'}$. If there are no unbounded feasible solutions, go to Step 6; otherwise go to Step 2.

Step 5: $Z^* = Z$; revise γ by putting $\gamma(1)$ at the end and moving all other elements up one position. Go to Step 7.

Step 6: Delete $\gamma(1)$ from γ and move all other elements up one position.

Step 7: Finished? If not, go to Step 2.

Sample Problem

Suppose the decision node order $\alpha = 3, 4, 1, 2, 5$ is specified the sample problem of Figure 3. Then the solution is as follows:

(1) $Z^* = \infty$

$\gamma = \{1\}$

$p_o^1 = \{\emptyset\}$

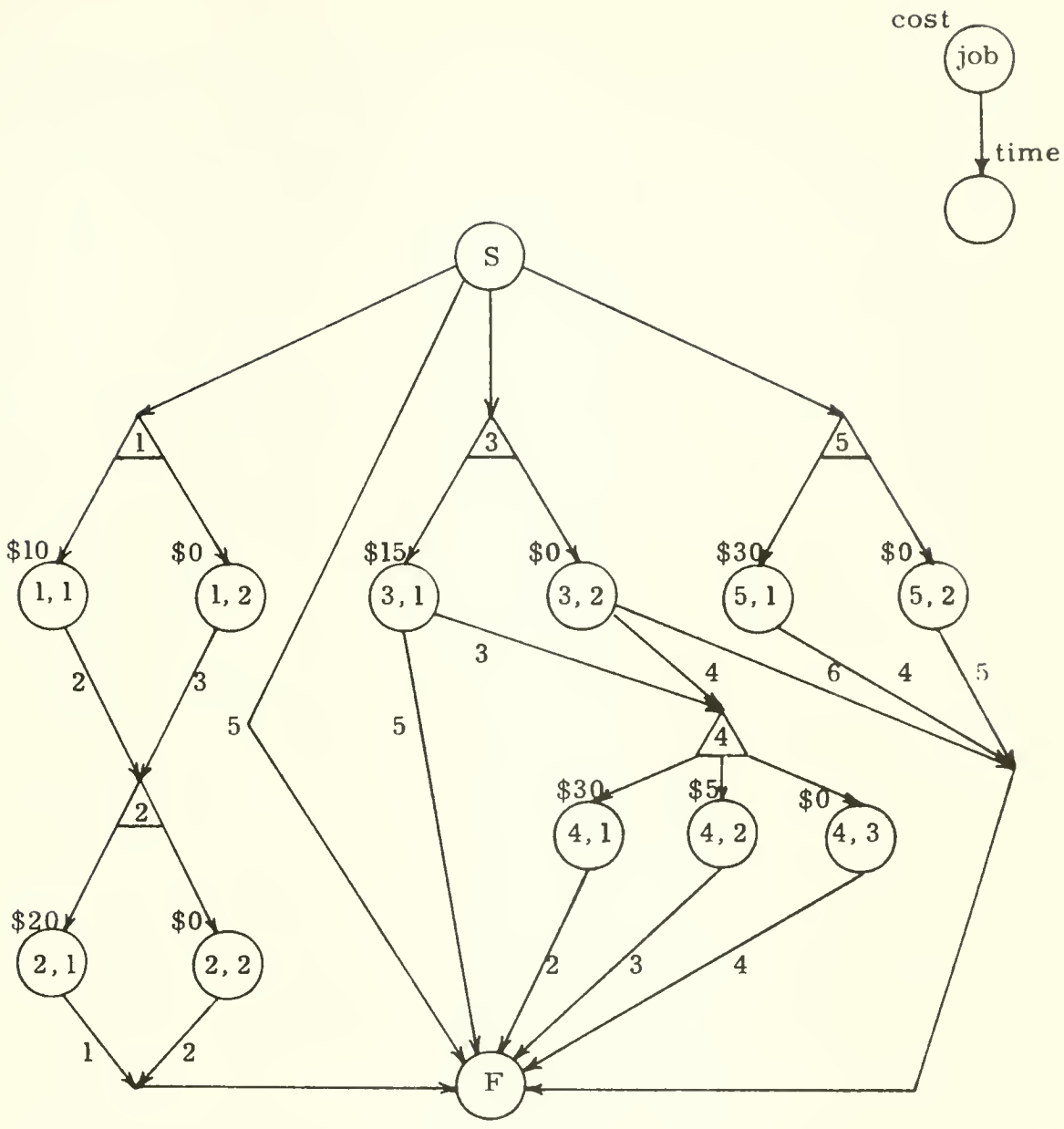
(2) $k = 1$; current solution is p_o^1

(3) $i = 3$

(4) $p_1^1 = \{d_{3,1}\}$ $w_F^1 = 5$ $C^1 = \$15$ $Z^1 = -\$5$

$p_1^2 = \{d_{3,2}\}$ $w_F^2 = 6$ $C^2 = \$0$ $Z^2 = \$40$

$\gamma = \{1, 2\}$ $Z^* = \infty$



DD = 6 days

r = \$20

p = \$40

Alternative Interdependency $d_{3,1} = d_{4,1}$

Figure 3. Sample Problem.

In the next iteration, solutions $\{d_{3,1}; d_{4,2}\}$ and $\{d_{3,1}; d_{4,3}\}$ are found to be infeasible. At the end of Step 4:

$p_2^1 = \{d_{3,1}; d_{4,1}\}$ $w_F^1 = 5$ $C^1 = \$45$ $Z^1 = \$45$

$p_1^2 = \{d_{3,2}\}$ $w_F^2 = 6$ $C^2 = \$0$ $Z^2 = \$40$

$\gamma = \{1, 2\}$ $Z^* = \infty$

The process continues until the only active solution is

$p_5^1 = \{d_{3,1}; d_{4,1}; d_{1,2}; d_{2,2}; d_{5,2}\}$ $w_F^1 = 5$

$C^1 = \$45$

$Z^1 = \$25$

The solution tree is shown in Figure 4.

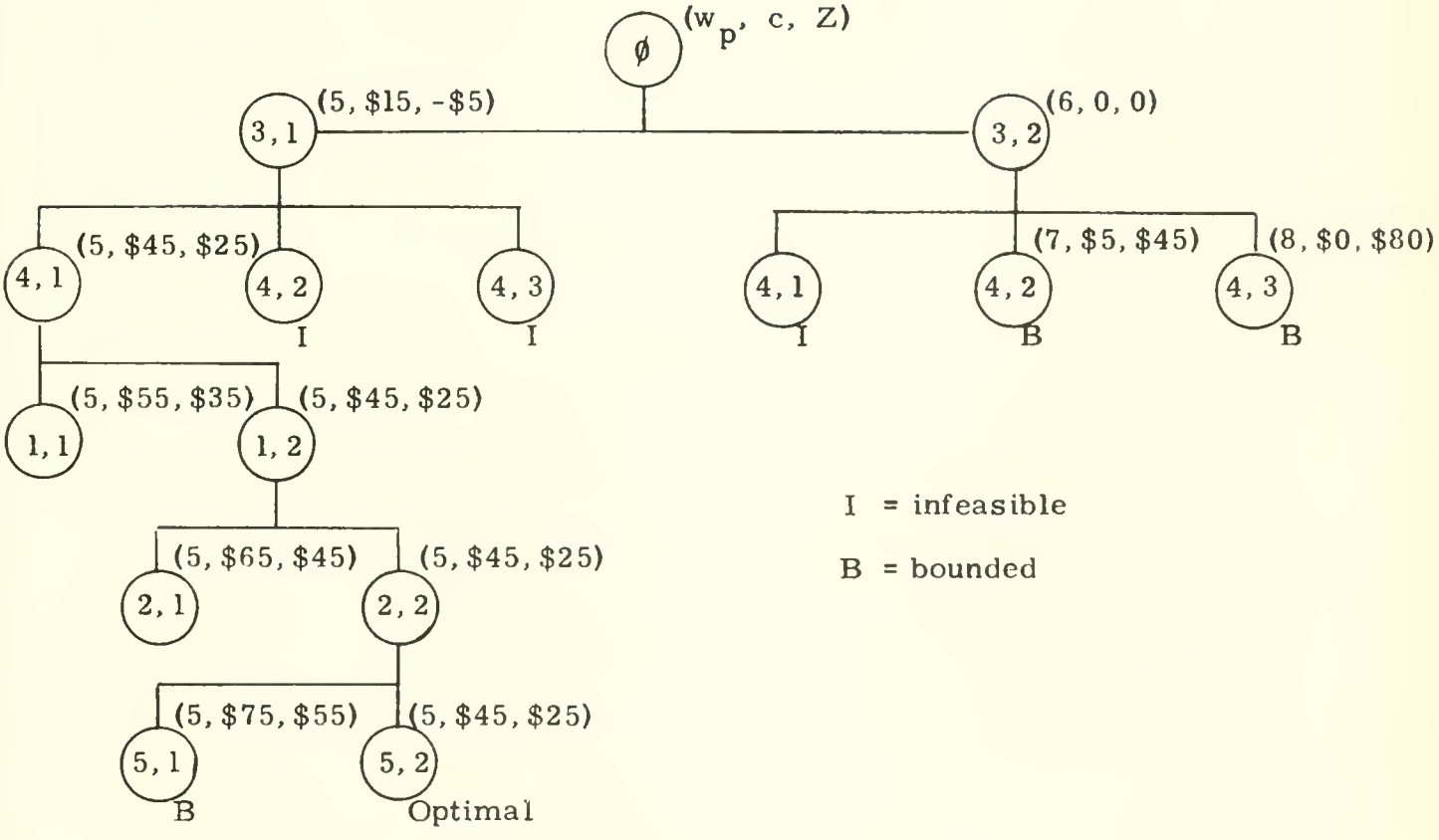


Figure 4.

Decision Node Order

Preliminary experimentation with the fixed order algorithm showed that solution time for a given problem was extremely sensitive to the order of decision nodes α. The major portion of this variability in solution time seemed to be concentrated in the time to prove optimality once the optimal solution had been found. It was further observed that solution time was minimized by those decision node orderings in which "critical" nodes (i.e., those nodes for which the chosen decision job in the optimal solution was on the critical path(s)) were processed first. With such orderings, strong w_F bounds were generated near the top of the solution tree, and the tree was consequently more effectively pruned.

In order to obtain a measure of the potential criticality of decision nodes, the following simple heuristic was devised: For each decision node choose the cheapest alternative. In the resulting network, calculate the total slack for each job. For a job not chosen, the calculated slack is that which would occur if the job were performed, but with the start times for the chosen jobs unaltered. In some cases, this implies negative slack. Choose the decision node order α on the basis of increasing $\min_j$ slack.[2] This ordering will be referred to as "slack order."

---

[2] This rule should perhaps be modified somewhat. Since the total slack calculated in this fashion is not particularly meaningful for those decision nodes not on the original critical path, it may be better to order these nodes so that they are grouped technologically. This gets more directly at the problem of ensuring that paths through the network are defined by successive decision nodes. The groups of nodes can be ordered on the basis of slack for the group. However, this is not a simple rule to apply for complex networks and has not been tested.

The algorithm was programmed in Fortran IV for an IBM 7094 computer and run under the time-sharing system of MIT's project MAC. The program was tested on variations of two fifteen-node, three-decisions-per-node (3^{15} possible solutions) problems. The two problem types, hereafter referred to as Type 1 and Type 2 problems, differed in the extent to which interconnections between opposite portions of the network existed. Type 2 problems contain many such interconnections, so that a number of competing critical paths were likely to develop. Typically, nine to eleven of the decision nodes in the problem became critical for some partial solution. Variety was provided to these problems by altering due dates, premiums, penalties, and times for precedence constraints (t_{ijkl}). Solution times are reported in Table 1, and do not include the three to five seconds required for input-output and initialization.

Results for the fixed order algorithm are shown for both slack order and technological order. Technological order offers some computational efficiencies in determining w_F^k but is essentially a random order with regard to potential criticality, and results have been included primarily for the basis of comparison. In all cases, slack order has proven to be superior to technological order.

It has been proposed that decision nodes involved in alternative interdependency constraints should be placed at the top of the decision tree so as to allow effective pruning on the basis of feasibility considerations. This, however, may interfere with a pure slack order, and results with a few test problems indicate that slack order should dominate.

Table 1. Comparison of Solution Times.

	Partitioning Tech. Order		Fixed Slack Order		Fixed Tech. Order	
1A	(2.8)	16.1	(6.4)	19.8	(T330)	---
1B	(11.3)	36.1	(16.4)	36.8	(73.7)	T213
1C	(1.6)	17.1	(13.8)	23.5	(51.0)	T157
1D	(1.7)	10.1	(101.3)	130.5	(139.4)	T261
1E	(2.4)	23.0	(12.8)	26.6	(7.7)	T141
2A	(1.0)	1.4	(1.8)	2.2	(14.3)	18.5
2B	(2.7)	5.3	(5.8)	9.4	(40.8)	73.2
2C	(0.8)	1.4	(2.0)	2.4	(2.2)	5.6
2D	(0.8)	1.0	(2.2)	2.4	(T97)	---
2E	(0.2)	0.4	(1.2)	1.6	(81.0)	T95

All times are in seconds. The times enclosed by parentheses are those required to find the optimal solution. The other times are those required to find and prove optimality. "T" indicates that the solution was prematurely terminated.

PARTITIONING ALGORITHM

Define the cheapest alternative completion Q_n^k of p_n^k to be the set of decisions d_{ij}, $i = \alpha(m)$, $m = n+1, \ldots, N$ feasible in conjunction with p_n^k such that $\sum_{m=n+1}^{N} C_{\alpha(m)j}$ is minimum. If $w_F^{k'}$ of $p_N^{k'} = p_n^k \cup Q_n^k$ is equal to w_F^k of p_n^k, then Q_n^k is clearly the best completion of p_n^k, and all

other completions of p_n^k need not be evaluated. The partitioning algorithm incorporates this consideration as follows:

Partition the decision nodes of a problem into two sets: B, the Branch and Bound set; and Q, the cheapest alternative set. Let b be the number of members in B. Solve the decision network defined by B with the fixed order algorithm. Each time a "complete" unbounded solution p_b^k to B is obtained, test $p_N^{k'} = p_b^k \cup Q_b^k$ to determine if $w_F^{k'} = w_F^k$. If not, transfer those decision nodes in Q which are critical into the set B and continue. Note that once a decision node enters B it remains there.

A simple version of the partitioning algorithm has been programmed in which all decision nodes involved in alternative interdependency constraints are placed in B. Then Q_b^k is simply the set of decisions $d_{ij} = 1$ such that $C'_{ij} = 0$, $i = \alpha(m)$, $m = n+1, \ldots, N$ and if $w_F^{k'} = w_F^k$, $Z^{k'} = Z^k$. More sophisticated versions of the partitioning algorithm can be envisioned in which the problem of determining Q_b^k is solved for the cases in which alternative interdependency constraints do affect the elements of Q. In many cases this may not be difficult. For the case in which it is possible to order decision nodes so that alternative interdependency constraints affect only successive nodes, the problem can be given a network interpretation and solved as a shortest-route problem.[1]

The partitioning algorithm is equivalent to the fixed order algorithm with the following amendment:

Step 2: $K = \gamma(1)$; current solution is p_n^k.
If $Z^k > Z^*$, go to Step 6. If $Z^k \leq Z^*$, go to Step 3 if $n < b$; Step 2A if $n = b$; and Step 5 if $n = N$.

Step 2A: Determine Q_b^k; $p_N^{k'} = p_p^k \; Q_p^k$, and $Z^{k'}$

Step 2B: If $w_F^{k'}$ of $p_N^{k'} = w_F^k$ of p_b^k, go to Step 5 with p_N. If not, go to Step 2C.

Step 2C: Determine critical path for $p_N^{k'}$. Place those nodes in Q which are critical into B. Go to Step 2.

Sample Problem

For the sample problem of Figure 3, decision nodes 3 and 4 are involved in alternative interdependency constraints, so $\alpha = \{3, 4, 1, 2, 5\}$ and b = 2. As before, at the end of two interations through Step 7:

$p_2^1 = \{d_{3,1} ; d_{4,1}\}$ $\quad w_F^1 = 5$ $\quad C^1 = \$45$ $\quad Z^1 = \$45$

$p_1^2 = \{d_{3,2}\}$ $\quad w_F^2 = 6$ $\quad C^2 = \$0$ $\quad Z^2 = \$40$

$\gamma = \{1, 2\}$ $\quad Z^* = \infty$

$B = \{3, 4\}$ $\quad b = 2$

$Q = \{1, 2, 5\}$

Returning to Step 2, the cheapest alternative completion, $Q_2^1 = S_{1,2} S_{2,2} S_{5,2}$, is found with p_2^1 to yield

$$p_5^1 = p_2^1 \cup Q_2^1 \quad d_{3,1}; d_{4,1}; d_{1,2}; d_{2,2}; d_{5,2}$$

$$w_F^1 \text{ of } p_5^1 = 5 = w_F^1 \text{ of } p_2^1$$

$$Z^1 \text{ of } p_5^1 = \$45 = Z^1 \text{ of } p_2^1$$

The complete solution tree is shown in Figure 5.

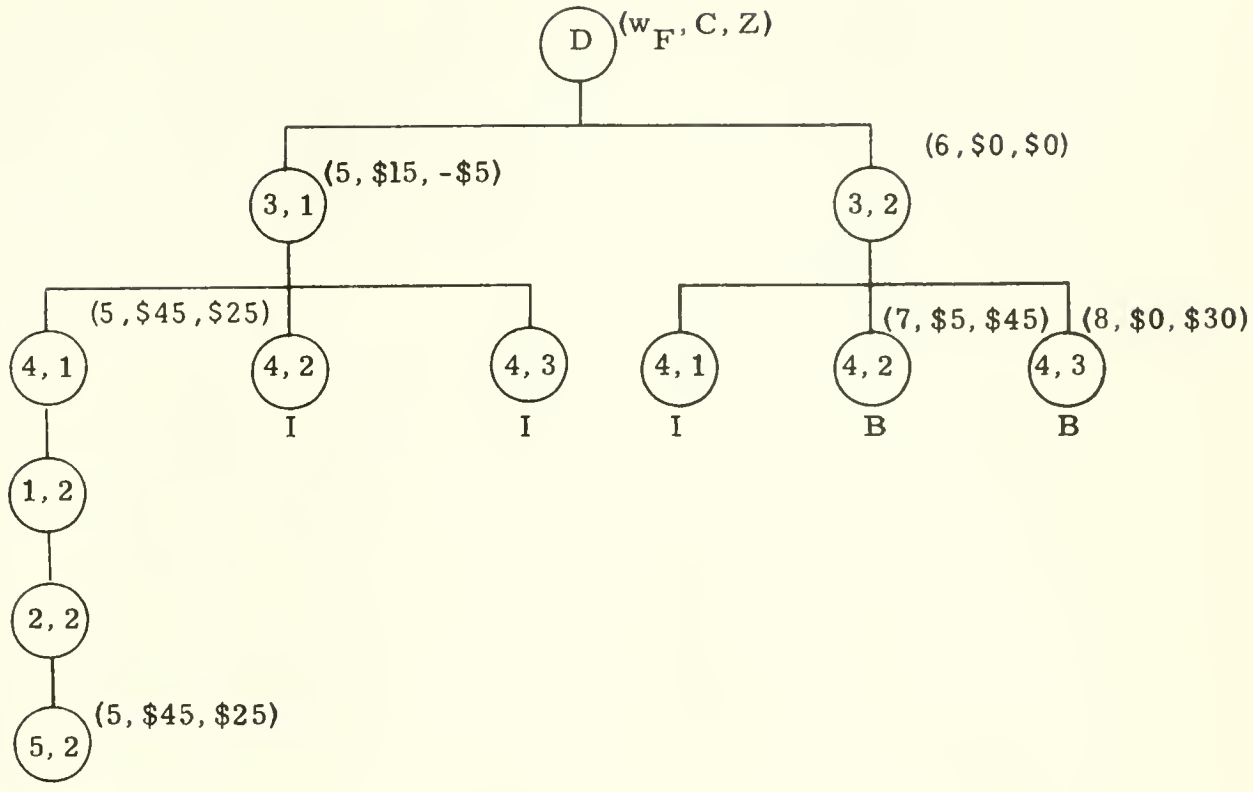


Figure 5.

Solution times for the partitioning algorithm are also shown in Table 1. In all cases, the algorithm performs better than the fixed order algorithm. The gains are slight for Type 1 problems because b, the size of the Branch and Bound set, grows to a large fraction of N (9/15 to 11/15). For Type 2 problems, improvements are of the order of 40 to 50 percent, for here b = 3 to 8. Although the major improvements shown occur for problems which are already easily solved, the partitioning algorithm may offer the potential for solving problems with very large N if b remains small.

Solution time for fixed N and k(i) is seen to be highly variable. It has been found that most of this variability can be explained by the final value of b, which is the measure of the number of nodes that become critical for some solution. The relation $\ell n t = a_0 + a_1 b$, where t is computer solution time, was fitted to the problems of Table 1 with the following results.

$$\begin{array}{lll} \ell nt = & 0.72 + & 0.47b \\ \text{t statistic} & (3.02) & (14.8) \end{array}$$

$F(1,8) = 218.2 \qquad \bar{R}^2 = 0.96$

$F(1,8)_{0.01} = 11.26 \qquad n = 10$

Although there was a significant relationship between these variables, it is not possible to predict computation time since b only becomes known during computation.

DECOMPOSITION

A large project may simply be a collection of relatively independent subprojects. If so, it may be possible to decompose the large networks into smaller subnetworks, "solve" the subnetworks, and then fit the subnetwork solutions together. In view of the exponential growth effect in solution time, it may be faster to solve a number of small problems in less time than to solve one large one. The idea is analogous to the decomposition principle for linear programming[3] and is related to the decomposition technique described by Parikh and Jewell in [7] for the time-cost tradeoff models of Kelly [4] and Fulkerson [5].

---

[3] See, for example, Ch. 22 and Ch. 23 in Dantzig [3].

A subproject is a collection of decision nodes in a reduced network which, taken together, "look like" a project. The collection must have a single starting point and a single finish. There can be no interdependency or precedence constraint between any node within the subproject and a node which is outside the subproject. The starting point may, however, be the successor of any number of jobs outside the subproject, and the finish may similarly be a predecessor of any number of jobs.

The enclosed portions of the network A of Figure 6 qualify as subprojects. The network is redrawn as in 6B with artificial start jobs (S', S") of zero cost and time inserted, as well as artificial finish points (F', F"). In Figure 6 the following is true:

(1) Every immediate successor of a subproject start variable is contained by the subproject boundary.

(2) Every immediate predecessor of a subproject finish variable is contained by the subproject boundary.

(3) For each subproject variable which is neither subproject start nor subproject finish, all immediate predecessors and successors are contained by the boundary.

An algorithm related to the network reduction routine of [1] has been developed to generate all legitimate subnetworks.

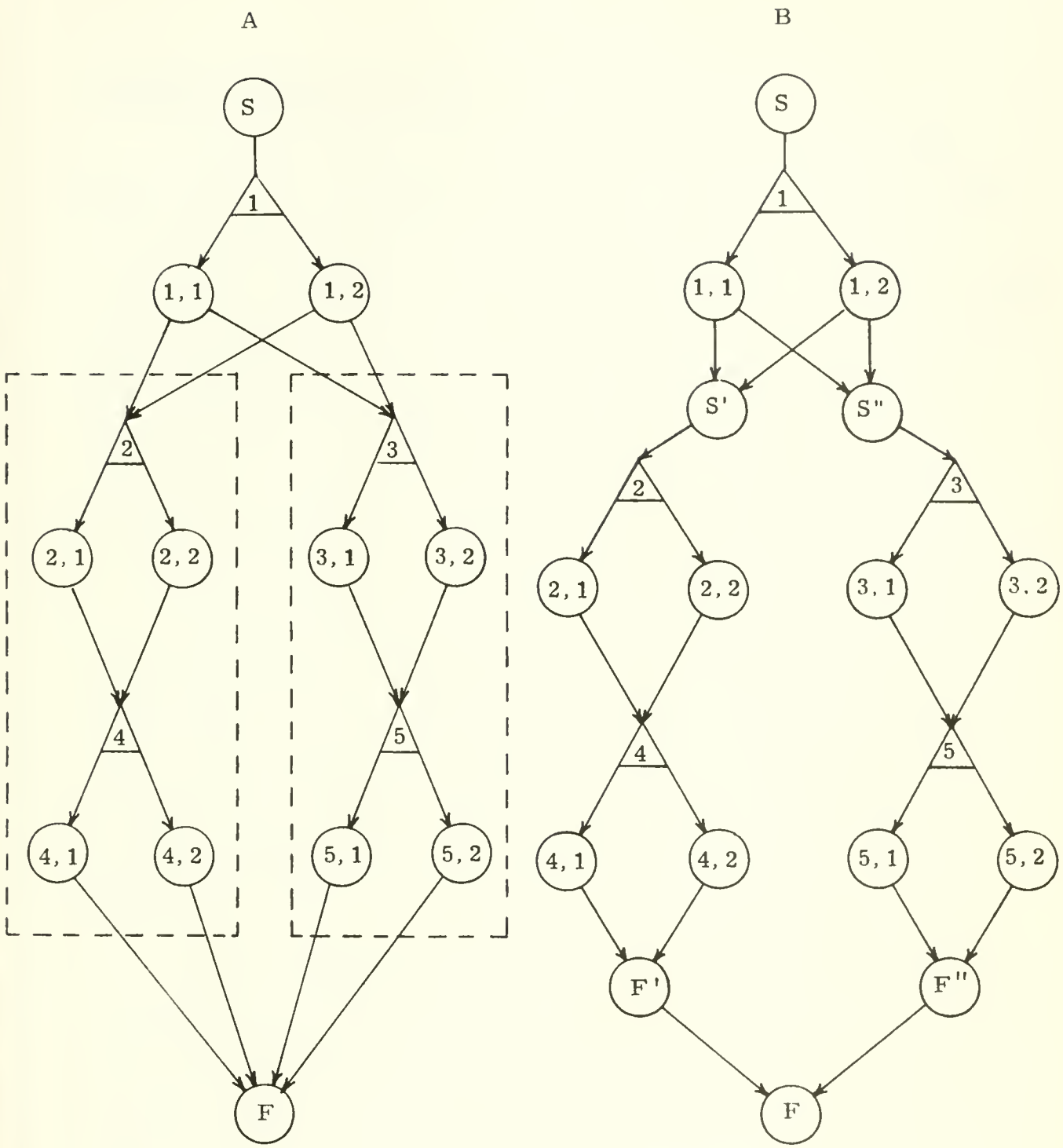


Figure 6. Decomposition of Project Networks

Combining Subproject Solutions

Due to the interaction between subprojects and the rest of the project network, it is generally not possible to solve for a single solution in each subproject and then simply to use these solutions to obtain an overall project optimum. A subproject does not have, in general, its own objective function. Rather, the objective function is concerned with the entire project completion date; and the effect of a subproject solution upon that date is not known until decisions are made for all other parts of the network. Consequently, some other method of fitting subproject solutions together in order to obtain the overall project optimum is needed.

The approach is based upon the observation that the set of all feasible solutions to a subproject has all of the characteristics of a decision node. Associated with each solution is a cost and a performance time, and each solution has a predecessor and a successor. In addition, only one solution may be chosen from the subproject -- i.e., the set of subproject solutions is subject to a mutually exclusive interdependency constraint.

Since each solution to a subproject has the same predecessor and successor (e.g., S' and F'), it is clear that the project optimal solution will never contain a subproject solution which is both longer and more expensive than some other feasible solution to the same subproject. The former subproject solution is "dominated" by the latter. In the case

where a subproject solution is less expensive than some shorter solution, the shorter solution is dominated if

$$C^2 - C^1 + (w_F^2 - w_F^1)\,(\max(p,r)) > 0$$

where C^1 and w_F^1 refer to the longer subproject solution, C^2 and w_F^2 to the shorter, and p and r are the overall project penalty and premium.

In order to obtain the optimal solution to the overall project, it is necessary to consider only all undominated solutions to each subproject. If all undominated solutions for a subproject have been obtained, the entire subproject can be replaced by a single decision node containing those solutions as its job alternatives. In the resulting master problem, there will be fewer decision nodes and fewer constraints. The master problem may then be solved with any of the Branch and Bound routines discussed in previous sections. Hopefully, the reduction in solution time for the master problem will be far greater than the time required to generate undominated solutions to the subprojects. The crucial issues as to the feasibility of this approach are: (1) the time required to generate all undominated solutions, and (2) the degree to which the space of all possible subproject solutions is reduced by dominance considerations.

The algorithm for determining all undominated solutions is a modified partitioning algorithm. Instead of a single Z^*, there now exists a $Z^*(t)$ for every subproject completion time (t). Initially, $Z^*(t) = \infty$ for all t.

Obtain a first complete solution with cost C' and completion time (t'). Then,

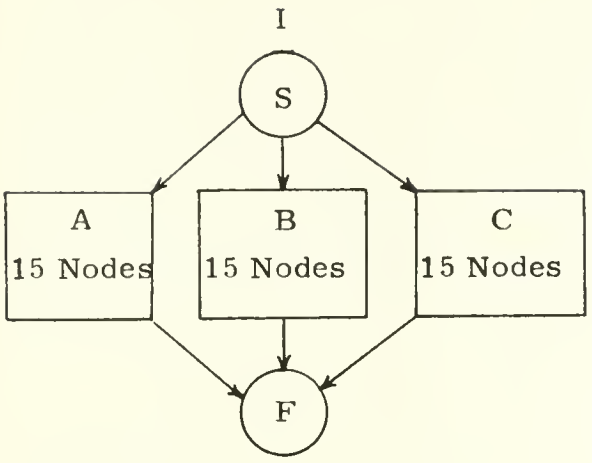
$$Z^*(t) = C' \text{ for all } t \geq t'$$

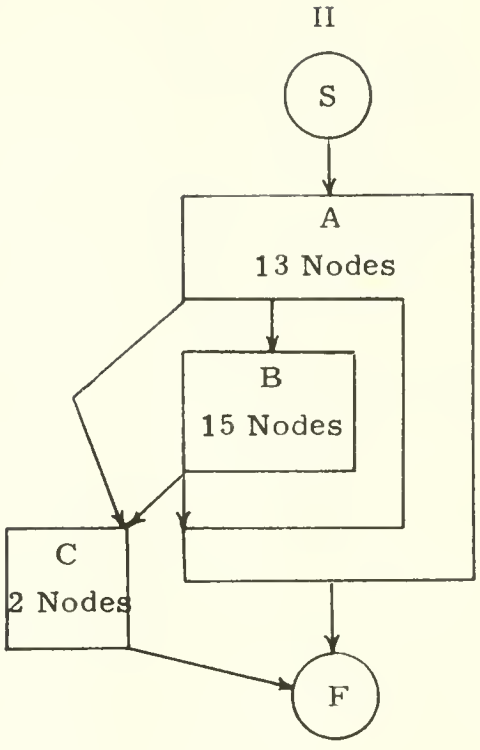
$$Z^*(t) = C' + [\max(p, r)](t' - t) \text{ for all } t < t'$$

When the next complete subproject solution is obtained, the same calculation is made for all t. $Z^*(t)$ is then equal to minimum of its current value and the new calculated value.

This algorithm was tested on the same sample of problems for which solution times are given in Table 1. It was found that only a small number of feasible undominated solutions were obtained (25 or less). This is to be expected -- there can be no more undominated solutions than the number of days separating the cheapest feasible solution and the shortest. More surprising is that the total amount of time required to generate all undominated solutions is in all cases less than three times the amount of time required to find the optimal solution(s). The conclusion that the determination of all undominated solutions will generally be at the same order of magnitude of difficulty as that of determining the optimal solution may be warranted. The partitioning algorithm, in searching for the optimal solution, is likely to generate a number of complete solutions which it eventually determines to be bounded; but these same solutions may be undominated and thus found by the undominated solution algorithm with very little additional effort.

To demonstrate the power of the decomposition approach, two networks were chosen (shown schematically in Figure 7). The master problem for Network I includes the undominated solutions from subprojects A, B, and C. The master problem of Network II includes the portion of the network A (not a subproject) and the undominated solutions from subprojects B and C.





With Decomposition

Subproject	Solution Time
A	3.8 sec
B	13.3 sec
C	3.4 sec
Master	1.8 sec
	22.3 sec

Without Decomposition

Total time: 163.4 sec

With Decomposition

Subproject	Solution Time
B	3.9 sec
C	0.8 sec
Master	35.1 sec
	39.8 sec

Without Decomposition

Total time: terminated

T 532 sec (optimal found but not proven)

Figure 7

Specialized techniques for solving the master problem may be incorporated in a fully automated decomposition algorithm. For example it may turn out that a subproject may always remain in the cheapest alternative set Q, in which case it is unnecessary to generate all undominated solutions for that subproject. Also, since the predecessor-successor relationships for all undominated solutions to a subproject are identical, the minimum time solution from each subproject may be used to strengthen the w_P bound in the master.

There are cases in which portions of a project network do not satisfy the strict requirements for designation as a subproject, but in which overall project optimum will be found by treating these portions as subprojects. Such will be the case when the precedence-successor relationships which violate the subproject requirement are not on the critical path in the optimal solution. Or they may be on the critical path, but the undominated solutions generated in the subproject may be undominated regardless of the existence of the critical links. Similarly, an alternative interdependency constraint may cross the boundary but have no effect on the optimal solution, or upon the undominated solutions generated by ignoring its existence. The case of the noncritical predecessor-successor link may often be easily seen ahead of time, so that the decomposition may proceed with an optimal solution guaranteed. In other cases the size of entire projects may be such that "illegal" decomposition may be resorted to in an effort to find a good, and hopefully optimal, solution. Also, since it is true that all alternatives of a decision node need not have identical predecessor and successor relations, it is conceivable that algorithms could be developed for reducing a segment of a network

to a decision node even though the segment does not meet the conditions of an "independent subproject."

CONCLUSION

Experimentation with the algorithms discussed in this paper shows that problem-solving time was strongly affected by the order in which the decision nodes were examined. Solution time was minimized by examining decision nodes in order of increasing "criticalness." The partitioning algorithm gave the lowest times for each problem tested. Its efficiency results from the fact that it tests each unbounded partial solution to a problem to see if the cheapest alternative for each decision node not included in the partial solution can be included without extending the project length. If this is the case, then all other completions of the partial solution need not be evaluated. The feasibility of the decomposition of large project networks was also examined. Independent subnetworks were defined within a large network and these were solved for all undominated solutions by an extension of the Partitioning Algorithm. The set of undominated solutions can then be regarded as a simple decision node within the large network and the resulting problem is solved by the partitioning algorithm described above. The improvement in solution time achieved by this decomposition was substantial.

REFERENCES

[1] Crowston, W. G., "Decision Network Planning Models", Management Science Research Report No.13B, Graduate School of Industrial Administration, Carnegie-Mellon U., 1968.

[2] Crowston, W. B., and G. L. Thompson, "Decision CPM: A Method of Simultaneous Planning, Scheduling, and Control of Projects", Operations Research, Vol.15, No.3, May-June 1967.

[3] Dantzig, G. E., Linear Programming and Extensions, Princeton University Press, Princeton, N.J., 1963, Chapters 22, 23.

[4] Fulkerson, D. R., "A Network Flow Computation for Project Activity Scheduling", Management Science, Vol. 7, 1961.

[5] Kelley, J. E., Jr., "Critical Path Planning and Scheduling: Mathematical Basis", Operations Research, Vol.7, No. 3, 1961.

[6] Lawler, E. L., and D. E. Wood, "Branch and Bound Methods: A Survey", Operations Research, Vol. 14, No. 4, 1966.

[7] Parikh, S. C., and W. S. Jewell, "Decomposition of Project Networks", Management Science, January 1965.

[8] Pierce, J. F., and D. J. Hatfield, "Production Sequencing by Combinatorial Programming", Operations Research and the Design of Management Information Systems (J. F. Pierce, Editor), Technical Association of the Pulp and Paper Industry, New York, 1967, p. 177.

[9] Wagner, M. H., "Solution of Decision CPM Networks", S.M. Thesis, Massachusetts Institute of Technology, June 1968.